Hit List

First Hit Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 1 of 1 returned.

☐ 1. Document ID: US 5761654 A

L4: Entry 1 of 1 File: USPT Jun 2, 1998

US-PAT-NO: 5761654
DOCUMENT-IDENTIFIER: US 5761654 A
**** See image for Certificate of Correction ****

TITLE: Memory structure and method for tuning a database statement using a join-tree data structure representation, including selectivity factors, of a master table and detail table

DATE-ISSUED: June 2, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tow; Daniel S.	Palo Alto	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 08/659158 [PALM]
DATE FILED: June 5, 1996

INT-CL-ISSUED: [06] G06 F 17/30

US-CL-ISSUED: 707/2; 707/3, 707/4, 707/101
US-CL-CURRENT: 707/2; 707/101, 707/3, 707/4

FIELD-OF-CLASSIFICATION-SEARCH: 395/602, 395/603, 395/604, 395/611, 707/2, 707/3, 707/4, 707/101
See application file for complete search history.

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5276870	January 1994	Shan et al.	395/600
5495605	February 1996	Cadot	395/600
5546570	August 1996	McPherson, Jr. et al.	395/600
5546571	August 1996	Shan et al.	395/600
5546576	August 1996	Cochrane et al.	395/600
5548755	August 1996	Leung et al.	395/600

5574900	November 1996	Huang et al.	395/601
5598559	January 1997	Chaudhuri	395/602
5608904	March 1997	Chaudhuri et al.	395/602
5619692	April 1997	Malkemus et al.	395/602
5625813	April 1997	Venn	395/602
5630120	May 1997	Vachey	395/602

OTHER PUBLICATIONS

Advertisement: "Operation SQL> Storm," Oracle Integrator, May/Jun. 1995; vol. 7, No. 3, p. 36.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Homere; Jean R.

ATTY-AGENT-FIRM: Arnold, White & Durkee

ABSTRACT:

A method of constructing an aid to tuning of database statements comprises a data structure (stored in a memory on a computer system) which compactly represents that information which is needed about a database statement to determine the optimal series of operations (e.g., table data fetches) needed to execute the statement. The information includes the relationships between tables joined in the statement and the selectivities of logical conditions applied to rows in those tables.

15 Claims, 8 Drawing figures

Term	Documents
MASTER	112748
MASTERS	11634
TABLE	899617
TABLES	204121
(3 AND (MASTER NEAR TABLE)).USPT.	1
(L3 AND (MASTER NEAR TABLE)).USPT.	1

Previous Page Next Page Go to Doc#

Refine Search

Search Results -

Term	Documents
MASTER	112748
MASTERS	11634
TABLE	899617
TABLES	204121
(3 AND (MASTER NEAR TABLE)).USPT.	1
(L3 AND (MASTER NEAR TABLE)).USPT.	1

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L4

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Tuesday, January 03, 2006 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L4	L3 and (master near table)	1	L4
L3	L2 and (primary near key) and (foreign near key)	38	L3
L2	L1 and (join near condition)	182	L2
L1	707/$.ccls.	16129	L1

END OF SEARCH HISTORY

L4: Entry 1 of 1 File: USPT Jun 2, 1998

DOCUMENT-IDENTIFIER: US 5761654 A
** See image for Certificate of Correction **
TITLE: Memory structure and method for tuning a database statement using a join-tree data structure representation, including selectivity factors, of a master table and detail table

Brief Summary Text (28):
Here, "A.fkey" represents a non-uniquely-indexed foreign key (in table.sub.-- A) pointing at the primary key for table.sub.-- B, i.e., "B.pkey". If the key is multi-column, there would be several of these equality conditions, of course.

Brief Summary Text (33):
Filter conditions are non-join conditions, which usually can be evaluated by looking at just a single table's columns. Rows that do not satisfy filter conditions are discarded as soon as the filter condition is evaluated, avoiding any further work on the rows, e.g., avoiding joining them to other tables later in the execution plan. Thus, as a general rule, it would be advantageous to evaluate non-join conditions first.

Brief Summary Text (40):
(2) for each directional link associated with a detail table and a master table, (i) the expected number of rows in the detail table that satisfy an average row in the master table, and (ii) the probability that a row in the detail table will have a corresponding row in the master table.

Detailed Description Text (5):
The data access statement in question specifies (i) a plurality of tables and (ii) a plurality of join conditions. Each of the join conditions specifies a relationship between a table that uses a key as a primary key, referred to as a master table, and a table that uses a corresponding key as a foreign key, referred to as a detail table.

Detailed Description Text (6):
As is well-known in the art, a primary key for a given table is a key that can take on a variety of values, but each individual value appears no more than once in that table. For example, in table 110 of FIG. 1B, the primary key for that table is the CUSTNUM key in column 114, because no two customers can have the same customer number.

Detailed Description Text (7):
A foreign key is a value that exists in a table and which references a primary key, usually in another table. For example, in table 110 of FIG. 1B, each customer's SALESREP value references a particular row in table 160 whose primary key is equal to that SALESREP value. That means that the SALESREP key is a foreign key in the customer table 110. A particular sales representative may represent more than one customer, which means that his or her SALESREP number may appear in more than one row of the customer table 110. In table 110, two customers, Barker and Lester, have CUSTNUM values 2030003 and 2030008 respectively. Their respective rows in table 110 include references to the same SALESREP value 2020. That value in the sales representative table corresponds to sales representative Earp. In the sales representative table 160, however, a given SALESREP number appears no more than once, because no two sales representatives have the same number.

Detailed Description Text (8):
Thus, in the context of the foregoing illustration, table 160 is a master table with respect to table 110, which in turn is a detail table with respect to table 160.

Detailed Description Text (9):
In the illustrative method, a set of nodes respectively representing the tables is defined, along with a set of directional links between pairs of nodes. Each directional link represents a master-detail relationship between a detail table and a corresponding master table. The specific directionality of the directional link is not crucial. In one embodiment, links may be defined as running from detail tables to corresponding master tables. For example, in FIG. 2, a link is depicted by an arrow running from detail table A to a master table B. If desired, of course, the links may run in the other direction, as long as consistent notation is maintained.

Detailed Description Text (11):
A representation of a set of properties of the nodes and links is associated with the join tree. The representation comprises (1) for each node, a set of zero or more selectivity factors; and (2) for each directional link associated with a detail table and a master table, (i) the expected number of rows in the detail table that satisfy an average row in the master table, and (ii) the probability that a row in the detail table will have a corresponding row in the master table.

Detailed Description Text (22):
When tuning an SQL statement, it can be useful to format the SQL statement in a fairly standardized way. In the example above, the keyword "from"is on a line by itself, so that the tables and aliases referenced can be found quickly. Each "table-name alias-name" pair is listed on its own line, making it easy to change the order of tables in the from clause. It is also useful to give each separate condition in the "where" clause its own line (if it will fit) so that join conditions and filters can be quickly identified.

Detailed Description Text (25):
Table aliases (or table names, if the tables are unaliased) are shown as nodes in a graph where joins are represented by links in the graph, with arrows on any side of the join representing a unique lookup, usually meaning an equi-join to a whole uniquely-indexed primary key. If exactly one side of the join points to a primary key (as is usually the case), the arrow points downward, toward the primary key. Next to the alias nodes of the graph, filters on those tables are represented in parentheses. If there were more than one filter on a table, each could be shown separately.

Detailed Description Text (26):
Alongside the arrow, multipliers *m and *f represent how many rows will join using the join condition in either direction. For instance, *m is the average of the number of rows in the detail table A that have foreign-key values that match a given primary-key value in table B. Phrased another way, *m is the ratio of (i) the number of distinct rows satisfying the join statement in detail table A to (ii) the number of distinct rows satisfying the join statement in master table B. If all rows in a detail table A have respective foreign keys matching respective primary keys in master table B, then *m is simply the number of rows in the detail table A divided by the number of rows having matching primary keys in the master table B.

Detailed Description Text (27):
As an example of *m, suppose that in FIG. 1B a simple join statement asks for a join of all rows in customer-number table 110 (a detail table) that have as a foreign-key value a sales-rep number used as a primary-key value in table 160. In this example, all 8 rows of table 110 satisfy the join condition, i.e., all 8 rows have a sales-rep number corresponding to one of the primary-key values used in table 160, and 4 distinct primary-key values from table 160 are used in the detail table 110. Thus, *m is the total number of rows in the detail table 110 (i.e., 8) divided by the total number of rows in master table 160 (i.e., 4), which is 2. If 5 distinct primary-key values from the master table 160 were used as foreign-key values in detail table A, then *m would be 8/5=1.6.

Detailed Description Text (28):
On the other hand, *f is the fraction of the number of rows in detail table A whose respective foreign-key values can also be found as primary-key values in master table B. In other words, *f may be thought of as the probability that a row in the detail table will have a corresponding or matching row in the master table. Since the join to B is through the primary key (guaranteed normally by a unique index), it is known that *f<=1. Usually, referential integrity and a "not null" condition on the foreign key further guarantee that the match from foreign key to primary key is successful, so usually *f=1, exactly. Where referential integrity is expected, the *f=.sub.-- condition can be omitted, implying that *f=1 by default.

Detailed Description Text (29):
Furthermore, because there is usually at least one detail row for every master row, it follows that m is usually >1, and A is usually a bigger table than B (in rowcount). In this common case, the arrow ".fwdarw." represents a many-to-one relationship in an entity relationship diagram. In the less-common case where detail rows may not exist, a (0-many)to-<something> relationship (and m may be <1 if the "0" case is common) exists. In the uncommon case where the foreign key may fail to find a matching primary key (no guarantee of referential integrity between these two tables), a <something>to-(0-1) relationship exists. It will be noted that the values for m and f are fixed properties of the relationship between the tables themselves, not of the query, as long as the query joins the tables through the normal full keys linking those tables.

Detailed Description Text (31):
The term "Cartesian product" is used here to indicate the result of generating all possible combinations of rows from two rowsets both having more than one row. A full Cartesian product results when two full multi-row tables are queried without any join condition at all linking them through primary or foreign keys.

Detailed Description Text (32):
Sometimes, some independent-looking filter condition can normalize a foreign-key-to-foreign-key join, discarding all but one row that is joined in one direction through the non-unique key, making that a virtual unique key when combined with the filter condition. For example, "deptno" in the classic EMPLOYEE table might be a virtual unique key if combined with a filter condition on a new, made-up column "union role" representing the role, if any, that employee plays in union representation:

Detailed Description Text (36):
Returning to FIG. 2, hypothetical numbers are now discussed to make the problem more concrete. Suppose that the rowcounts for A and B are ascertained, and that A has three times as many rows as B. Generally, that would make *m=3, unless the detail table A sometimes has details that do not join to the master table B. Further assume referential integrity between these tables, so *m=3 and *f=1.

Detailed Description Text (37):
(The rowcounts may be obtained with "select count(*) from <table>" for each table. Alternatively, the Oracle iq.sql script may be used to see the number of primary-key-index distinct keys (which is the same as the rowcount as of the last time the table or index was analyzed), as long as the table or primary key index has been analyzed since the primary key index was last created. For big tables, iq.sql would run much faster than selecting count(*) from the whole table, which requires a full table scan. In Oracle Applications, the primary key index is named <table.sub.-- name>.sub.-- U1 by convention.) The iq.sql1 script is as follows:

Detailed Description Text (84):
Assume that indexes exist for both A.effective_date and B.phase and for both the foreign and primary key involved in the join. Both tables have filters that are selective enough (<=0.5% of rows) that use of a full table scan normally would not be considered (given a choice) if that table were being queried for a single-table fetch (see above). It turns out that the same criteria apply to full table scans for a multi-table join, so it will very likely be desirable to follow a nested-loops join, entering the first ("driving") table through the index on its filter condition.

Detailed Description Text (86):
For concreteness, assume that table B has 1,000,000 rows, and that the cost of the selected execution plan is simply the sum of the number of rows fetched from each table, with no need to correct for how well cached each table is, or how deep each index is. (It is very rare that such correction alters which plan is best.) All that is needed to compare the two remaining possibilities is now known--enter A through the index on effective.sub.-- date and drive to B with the index on the primary key, or alternatively, enter B with the index on phase and drive to A with the index on the foreign key. The cost can be calculated as follows:

Detailed Description Text (87):
Driving from table A, that table must contain about 3*1,000,000 rows (because B has 1,000,000 rows and m=3), and the filter on effective.sub.-- date will result in just (0.005)*(3)* (1,000,000) rows from A. Each of those rows will join to exactly one row from B (through the primary key), so the total number of rows fetched is

Detailed Description Text (89):
It should be noted that for finding the cost of this plan, the selectivity of the filter on B is irrelevant, because the filter is only applied after the rows from B are fetched. The index on B.phase should not be used if driving from A--in general, indexes on filter conditions should only be used for driving tables. (The index on B.phase could be used, but normally would not be a good idea. This index could be merged with the unique index on B.pkey, but it is ordinarily much cheaper to simply fetch a row from the table, even if it is a row that is discarded, than it would be to merge the one rowid from B.pkey with the 2000 rowids from A.phase before going to the table. Alternatively, the index on B.phase could be used instead of the index on B.pkey, but that would fetch 2,000 times more rows; more than 1,999 of the 2,000 rows just fetched would then be discarded because they failed to satisfy the join condition.)

Detailed Description Text (91):
This last analysis has a subtle built-in assumption that is usually true: Starting with a fraction "F" of rows from a master table, a join will link those rows to roughly a fraction "F" of the rows in the detail table. This assumed behavior of joins is referred to as "inheritance of filter selectivities by detail tables." Thus, a filter on headers selecting 1% of the headers should result in a join to about 1% of the lines, and 3% of the customers should join to about 3% of the orders.

Detailed Description Text (95):
and it would seem that the condition on lookup.sub.-- type may be a fairly selective filter (perhaps F=0.01). Yet, for purposes of estimating how many detail rows will be joined, it is essentially no filter at all--all rows in detail.sub.-- table have a reserve.sub.-- id pointing back to a lookup.sub.-- code with that lookup.sub.-- type. A better way to view this case (which also avoids the problem of failing to have a primary key on either side of the join) is to see the "filter" on lookup.sub.-- type as no filter at all, but instead as part of the primary key to that table, only to be used in tandem with the rest of the key.

Detailed Description Text (115):
A few more complexities will be discussed below before moving beyond two-way joins. Sometimes, the direction of a join is unavoidable, preventing an execution plan starting from the preferred filter. For example, an outer join forces the plan to drive "toward the (+)." While the existence of an index on a primary key can almost always be assumed, not all foreign keys have (or even should have) indexes. Consequently, joins going "upstream" on the diagram only work (using the usually-preferred nested loops plan) if that foreign key is indexed.

Detailed Description Text (116):
By convention, such enforced join directions will be shown with a small directional carat embedded in the join arrow. For example, if the join condition in our example is changed to "and A.fkey(+)=B.pkey," the diagram would be as shown in FIG. 5. In that case, it would probably be necessary to drive from the filter on B whether that was desirable or not. Alternatively, if the outer join was written: "and A.fkey=B.pkey(+)," or if no index on A.fkey existed, the diagram would be as shown in FIG. 6, and it would probably be necessary to drive

from the filter on A even though it would be preferable to drive from the filter on B.

Detailed Description Text (126):
Nested-loops joins are the most common in Oracle Applications SQL. A nested-loops join of A to B is carried out by fetching rows from the driving table A and for each of these rows finding the matching row(s) in B (in a loop executed for each row in A that passes the filters on A), usually using the index on a primary or foreign key that points back at A. In the case of many-way joins, a later nested loop will join to a table X finding the matching rows in X for each row so far generated from the earlier joins (and passing all filter conditions applicable up until that point), usually through an index on a primary or foreign key pointing back to any one or more of the earlier-joined tables pointed to by that key.

Detailed Description Text (127):
Occasionally, nested loops are found that join to tables through an index that is on a filter condition or on only part of a key, or even that joins through nested loop to a full table scan. These almost always cause problems, fetching many times more rows from the joined-to table than would have been fetched by using the full join key. For example, a driving table returning just 10 rows can cause 10 full table scans if the next table is accessed by full table scan. Nested loops to anything other than a full primary or foreign key are likely to result in a Cartesian product of two potentially large rowsets.

Detailed Description Text (138):
1. either the driving table is forced not to be the table with the best filter (owing to lack of a foreign key index or owing to an outer join), or the table with the best set of filter conditions drives to a large enough fraction of the second table that a full table scan on the second table is preferred; and

Detailed Description Text (145):
To illustrate how the approach extends to more complicated queries, the tuning of the following hypothetical 8-way join, which has 8!=40,320 possible join orders, is explained below. The primary and foreign keys for most of these joins have two columns: First, the org.sub.-- id column; and second, the <p/f>key<n> column:

Detailed Description Text (147):
Achieving a tuned execution plan requires a systematic approach. Assume it is known that indexes exist on all the foreign keys referenced except B.fkey6, which is the foreign key pointing to table C. All tables except the lookup table H have two-part primary keys, (org.sub.-- id, pkey<n>), uniquely indexed in that order. (Incidentally, it is usually not a good idea to have indexes start with an unselective column like org.sub.-- id.) The primary key to the lookup table H is uniquely indexed on the columns (H.lookup.sub.-- type, H.lookup.sub.-- code) in that order.

Detailed Description Text (151):
Assume first that *m is high, i.e., table D has many more rows than table B. Also note that the conditions of org.sub.-- id are part of the join condition for all but the driving table. Now assume a decision is made to rank the remaining filters as follows:

Detailed Description Text (160):
(4) Repeat until all tables are joined through full primary or foreign keys.

Detailed Description Text (169):
2. Occasionally, drive from another table if it also has a very good filter that uses an index, where the better combination of filters cannot use an index selectively. (For example, the better filter, which might nevertheless not be used, might be a very selective "is null" condition or a condition on an unindexed column.) Rarely, it may be necessary to drive from an inferior filter because the best filter cannot access the rest of the join tree owing to an outer join or owing to lack of an index on a foreign key. In this case, it is desirable to start as close to the best filter as possible while still having access to the rest of the tree. (In the example above, the driving would begin at table D to get to the filter on table A, and from table B to get to the filter on table C if either A or C had the best filter,

because it is not possible to access the rest of the join tree from either table A or table C.)

Current US Original Classification (1):
707/2

Current US Cross Reference Classification (1):
707/101

Current US Cross Reference Classification (2):
707/3

Current US Cross Reference Classification (3):
707/4

CLAIMS:

1. A machine-executed method of creating a data structure in a memory device, for use in selecting an execution plan for a data access statement that specifies (i) a plurality of tables and (ii) a plurality of join conditions, each of said join conditions specifying a relationship between (x) a table that uses a key as a primary key, referred to as a master table, and (y) a table that uses a corresponding key as a foreign key, referred to as a detail table, said method comprising:

(a) defining a set of nodes respectively representing the tables;

(b) defining a set of directional links between pairs of nodes each of which represents a master-detail relationship between a detail table and a corresponding master table;

(c) defining, in a memory device, a data structure, referred to as a join tree, comprising a representation of the nodes and their directional links;

(d) associating with the join tree a representation of a set of properties of the nodes and links, comprising:

(1) for each node, a set of zero or more selectivity factors, each selectivity factor indicating the expected fraction of rows in the table represented by the node that satisfy one or more logical conditions set forth in the data access statement;

(2) for each directional link associated with a detail table and a master table, (A) the ratio of (i) the number of distinct rows satisfying the join statement in the detail table A to (ii) the number of distinct rows satisfying the join statement in the master table, and (B) the probability that a row in the detail table will have a corresponding row in the master table.

4. A machine-executed method of creating a data structure in a memory device for use in selecting an execution plan for an SQL statement that specifies (i) a plurality of tables and (ii) a plurality of join conditions each of which specifies a relationship between (x) a table that uses a key as a primary key, referred to as a master table, and (y) a table that uses a corresponding key as a foreign key, referred to as a detail table, said method comprising:

(a) defining a set of nodes respectively representing the tables;

(b) defining a set of directional links between pairs of nodes each of which represents a master-detail relationship between a detail table and a corresponding master table;

(c) defining, in said memory device, a data structure, referred to as a join tree, comprising a representation of the nodes and their directional links;

(d) associating with the join tree a representation of a set of properties of the nodes and links, comprising:

(1) for each node, a set of zero or more filters, each filter indicating the expected fraction of rows in the table represented by the node that satisfy one or more logical conditions set forth in the SQL statement;

(2) for each directional link associated with a detail table and a master table, (A) the ratio of (i) the number of distinct rows satisfying the join statement in the detail table A to (ii) the number of distinct rows satisfying the join statement in the master table, and (B) the probability that a row in the detail table will have a corresponding row in the master table.

7. A machine-readable memory device encoding a map of a data access statement, said data access statement specifying (i) a plurality of tables and (ii) a plurality of join conditions, each of said join conditions specifying a relationship between (x) a table that uses a key as a primary key, referred to as a master table, and (y) a table that uses a corresponding key as a foreign key, referred to as a detail table, said map comprising respective representations of:

(a) a set of nodes respectively representing the tables;

(b) a set of directional links between pairs of nodes, each directional link representing a master-detail relationship between a detail table and a corresponding master table;

(c) a set of properties of the nodes and links, comprising:

(1) for each node, a set of zero or more selectivity factors, each selectivity factor indicating the expected fraction of rows in the table represented by the node that satisfy one or more logical conditions set forth in the data access statement;

(2) for each directional link associated with a detail table and a master table, (A) the ratio of (i) the number of distinct rows satisfying the join statement in the detail table A to (ii) the number of distinct rows satisfying the join statement in the master table, and (B) the probability that a row in the detail table will have a corresponding row in the master table.

10. A machine-readable memory device encoding a map of an SQL statement, said SQL statement specifying (i) a plurality of tables and (ii) a plurality of join conditions, each of said join conditions specifying a relationship between (x) a table that uses a key as a primary key, referred to as a master table, and (y) a table that uses a corresponding key as a foreign key, referred to as a detail table, said map comprising respective representations of:

(a) a set of nodes respectively representing the tables;

(b) a set of directional links between pairs of nodes, each directional link representing a master-detail relationship between a detail table and a corresponding master table;

(d) a set of properties of the nodes and links, comprising:

(1) for each node, a set of zero or more filters, each filter indicating the expected fraction of rows in the table represented by the node that satisfy one or more logical conditions set forth in the SQL statement;

(2) for each directional link associated with a detail table and a master table, (A) the ratio of (i) the number of distinct rows satisfying the join statement in the detail table A to (ii) the number of distinct rows satisfying the join statement in the master table, and (B) the probability that a row in the detail table will have a corresponding row in the master table.

Previous Doc Next Doc Go to Doc#